# Dynamical Systems CW4

CID: 02044064

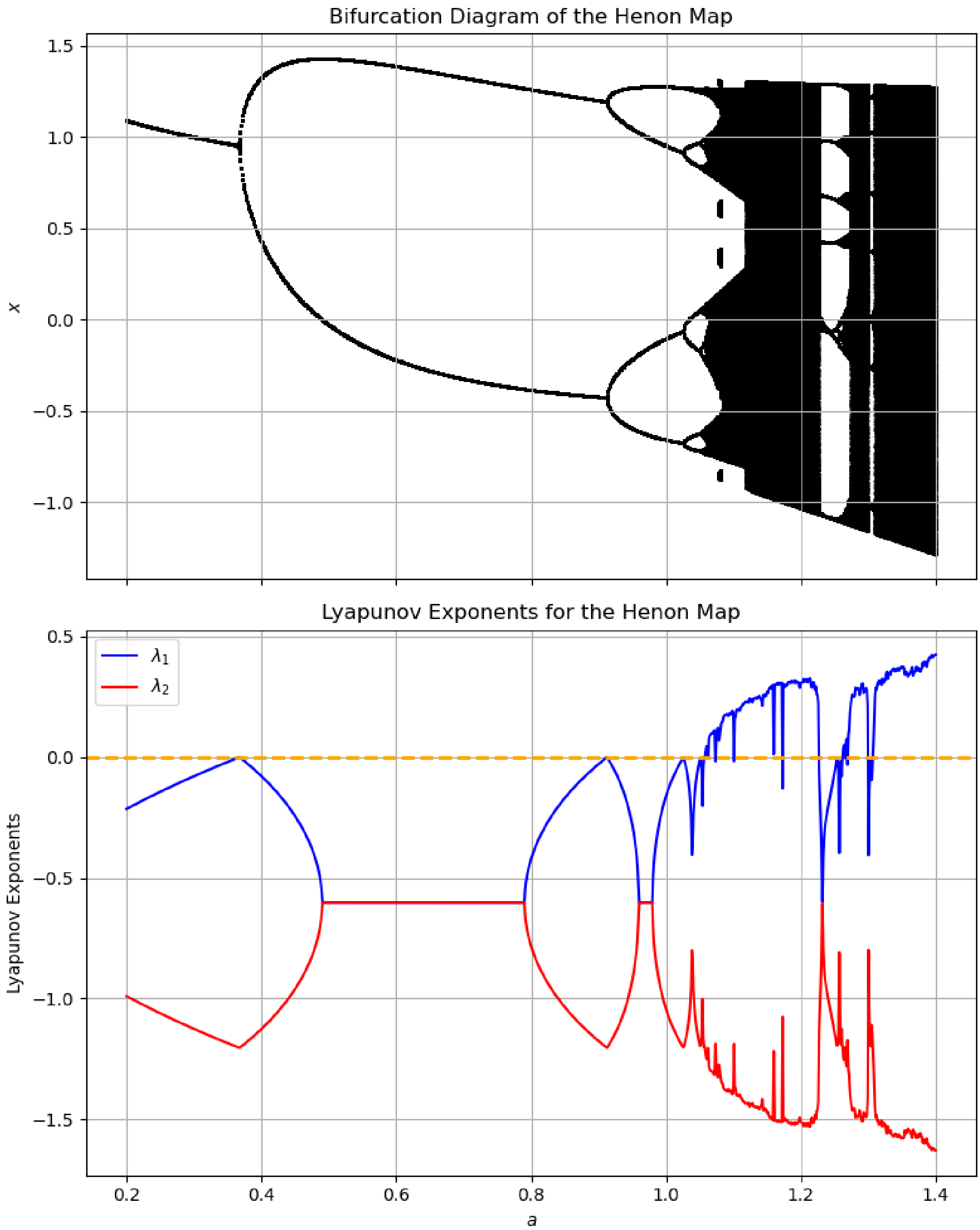
February 2025

## Q1

```
import numpy as np
import matplotlib.pyplot as plt


# Part a: Computing the Lyapunov exponents


def gram_schmidt(vectors):
    dim = vectors.shape[1]
    ortho_vectors = np.copy(vectors)
    norms = np.zeros(dim)

    for i in range(dim):
        for j in range(i):
            proj = np.dot(ortho_vectors[:, j], ortho_vectors[:, i]) * ortho_vectors[:, j]
            ortho_vectors[:, i] -= proj
        norms[i] = np.linalg.norm(ortho_vectors[:, i])
        ortho_vectors[:, i] /= norms[i]

    return ortho_vectors, norms


def henon_map(x, y, a, b):
    return 1 - a * x ** 2 + y, b * x


def jacobian(x, y, a, b):
    return np.array([[-2 * a * x, 1], [b, 0]])


def compute_lyapunov_exponents(a_values, b=0.3, steps=5000, discard=1000):
    lyapunov_exp_1 = []
    lyapunov_exp_2 = []

    for a in a_values:
        x, y = 0, 0  # Initial condition
        Q = np.eye(2)  # Orthonormal basis
        lyap_sum = np.zeros(2)

        for i in range(steps):
            x, y = henon_map(x, y, a, b)
```

```python
            J = jacobian(x, y, a, b)
            Q = J @ Q  # Left multiply basis vectors by Jacobian
            Q, R = gram_schmidt(Q)  # Reorthogonalise

            if i >= discard:
                lyap_sum += np.log(np.abs(R))

        # Take average
        lyapunov_exp_1.append(lyap_sum[0] / (steps - discard))
        lyapunov_exp_2.append(lyap_sum[1] / (steps - discard))

    return np.array(lyapunov_exp_1), np.array(lyapunov_exp_2)


# Part b: Bifurcation diagram


def bifurcation_diagram(ax, a_values, b=0.3, steps=5000, discard=1000):
    x_vals = []
    a_vals = []

    for a in a_values:
        x, y = 0, 0  # Initial condition

        for i in range(steps):
            x, y = henon_map(x, y, a, b)
            if i >= discard:
                x_vals.append(x)
                a_vals.append(a)

    ax.scatter(a_vals, x_vals, s=0.1, color='black')
    ax.set_ylabel('$x$')
    ax.set_title('Bifurcation Diagram of the Henon Map')
    ax.grid()


# Part c: Plots


a_values = np.linspace(0.2, 1.4, 1000)
lambda1, lambda2 = compute_lyapunov_exponents(a_values)

fig, axs = plt.subplots(2, 1, figsize=(8, 10), sharex=True)

# top subplot
bifurcation_diagram(axs[0], a_values)

# bottom subplot
axs[1].plot(a_values, lambda1, label='$\lambda_1$', color="blue")
axs[1].plot(a_values, lambda2, label='$\lambda_2$', color='red')
axs[1].axhline(0, color='orange', linestyle="dashed", linewidth=2)
axs[1].set_xlabel('$a$')
axs[1].set_ylabel('Lyapunov Exponents')
axs[1].set_title('Lyapunov Exponents for the Henon Map')
```

```
axs[1].legend()
axs[1].grid()

plt.tight_layout()
plt.show()
```




My code is largely based on lectures 7 (Gram-Schmidt) and 10 (Lyapunov exponents calculation and plot, as well as the bifurcation diagram).